# Babel 和其他工具

一旦你掌握的窍门，安装 Babel 还是十分简明的，不过和其他工具搭配在一起就会变得困难多了。不过我们一直在与其他项目密切合作以确保这种体验尽可能简单。

# 静态分析工具

新标准为语言带来了许多新的语法，静态分析工具正在将此利用起来。

## 语法检查（Linting）

ESLint 是最流行的语法检查工具之一，因此我们维护了一个官方的 `babel-eslint` 整合软件包。首先安装 `eslint` 和 `babel-eslint`。.

```
$ npm install --save-dev eslint babel-eslint
```

**注意：**兼容 Babel 6 的 `babel-eslint` 目前正处于预发行版本。 安装最新的 5.0 beta 版来兼容 Babel 6。

然后创建或使用项目现有的 `.eslintrc` 文件并设置 `parser` 为 `babel-eslint`。.

```
 {
+  "parser": "babel-eslint",
   "rules": {
     ...
   }
 }
```

现在添加一个 `lint` 任务到 npm 的 `package.json` 脚本中：

```
 {
   "name": "my-module",
   "scripts": {
+    "lint": "eslint my-files.js"
   },
   "devDependencies": {
     "babel-eslint": "...",
     "eslint": "..."
   }
 }
```

接着只需要运行这个任务就一切就绪了。

```
$ npm run lint
```

详细信息请咨询 `babel-eslint` 或者 `eslint` 的文档。

# 代码风格

JSCS 是一个极受欢迎的工具，在语法检查的基础上更进一步检查代码自身的风格。 Babel 和 JSCS 项目的核心维护者之一（@hzoo）维护着 JSCS 的官方集成。

更妙的是，JSCS 自己通过 `--esnext` 选项实现了这种集成，于是和 Babel 的集成就简化成了直接在命令行运行：

```
$ jscs . --esnext
```

或者在 `.jscsrc` 文件里添加 `esnext` 选项。

```
  {
    "preset": "airbnb",
+   "esnext": true
  }
```

详细信息请咨询 `babel-jscs` 或是 `jscs` 的文档。

# 文档

使用 Babel，ES2015，还有 Flow 你可以对你的代码进行大量的推断。使用 documentation.js 可以非常简便地生成详细的 API 文档。

Documentation.js 使用 Babel 来支持所有最新的语法，包括用于在你的代码中声明类型所用的 Flow 注解在内，

# 框架

所有主流的 JavaScript 框架都正在努力调整他们的 APIs 向这门语言的未来看齐。有鉴于此，配套工具方面已经做出了大量的工作。

除了使用 Babel 以外，框架更有条件去扩展 Babel 来帮助他们提升用户体验。

# React

React 已经大幅改变了他们的 API 以适应 ES2015 的类语法（此处了解更新的 API）。 特别是 React 现在依赖 Babel 编译它的 JSX 语法且弃用了它原有的自定义工具。 你可以按照上述说明安装 `babel-preset-react` 包来开始。.
React 社区采用 Babel 并围绕它来运行，现在社区已经创建了大量的转换器（transforms）。.

最令人瞩目的是 `babel-plugin-react-transform` 插件，它集成了大量 React 专用转换器可以启用诸如 *热模块重载*等其他调试工具。